**--1. Customers Table**

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone VARCHAR(15),

registered\_date DATE

);

INSERT INTO Customers (first\_name, last\_name, email, phone, registered\_date) VALUES

('Alice', 'Johnson', 'alice.johnson@example.com', '1234567890', '2022-01-10'),

('Bob', 'Smith', 'bob.smith@example.com', '1234567891', '2022-02-15'),

('Charlie', 'Brown', 'charlie.brown@example.com', '1234567892', '2022-03-20'),

('David', 'White', 'david.white@example.com', '1234567893', '2022-04-05'),

('Emma', 'Wilson', 'emma.wilson@example.com', '1234567894', '2022-05-12'),

('Frank', 'Thomas', 'frank.thomas@example.com', '1234567895', '2022-06-22'),

('Grace', 'Taylor', 'grace.taylor@example.com', '1234567896', '2022-07-18'),

('Hannah', 'Anderson', 'hannah.anderson@example.com', '1234567897', '2022-08-09'),

('Isaac', 'Martinez', 'isaac.martinez@example.com', '1234567898', '2022-09-30'),

('Jack', 'Lopez', 'jack.lopez@example.com', '1234567899', '2022-10-21');

**--2. Products Table**

CREATE TABLE Products (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100),

category VARCHAR(50),

price DECIMAL(10,2)

);

INSERT INTO Products (product\_name, category, price) VALUES

('Laptop', 'Electronics', 1200.00),

('Headphones', 'Electronics', 150.00),

('Smartphone', 'Electronics', 800.00),

('Monitor', 'Electronics', 300.00),

('Keyboard', 'Electronics', 50.00),

('Mouse', 'Electronics', 30.00),

('Chair', 'Furniture', 200.00),

('Table', 'Furniture', 300.00),

('Sofa', 'Furniture', 700.00),

('Bed', 'Furniture', 1000.00);

**-- 3. Orders Table**

CREATE TABLE Orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10,2),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

INSERT INTO Orders (customer\_id, order\_date, total\_amount) VALUES

(1, '2023-05-10', 1350.00),

(2, '2023-06-15', 500.00),

(3, '2023-07-20', 900.00),

(4, '2023-08-25', 1200.00),

(5, '2023-09-30', 450.00),

(6, '2023-10-05', 700.00),

(7, '2023-11-11', 1500.00),

(8, '2023-12-20', 600.00),

(9, '2024-01-15', 1800.00),

(10, '2024-02-05', 950.00);

**-- 4. Order Details Table**

CREATE TABLE Order\_Details (

order\_detail\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

product\_id INT,

quantity INT,

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

INSERT INTO Order\_Details (order\_id, product\_id, quantity) VALUES

(1, 1, 1), (1, 2, 1), (2, 3, 2), (3, 4, 3), (4, 5, 2),

(5, 6, 1), (6, 7, 2), (7, 8, 1), (8, 9, 3), (9, 10, 1);

**-- 5. Payments Table**

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

payment\_date DATE,

payment\_method VARCHAR(20),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id)

);

INSERT INTO Payments (order\_id, payment\_date, payment\_method) VALUES

(1, '2023-05-11', 'Credit Card'),

(2, '2023-06-16', 'PayPal'),

(3, '2023-07-21', 'Bank Transfer'),

(4, '2023-08-26', 'Credit Card'),

(5, '2023-10-01', 'PayPal'),

(6, '2023-10-07', 'Credit Card'),

(7, '2023-11-12', 'Bank Transfer'),

(8, '2023-12-21', 'PayPal'),

(9, '2024-01-16', 'Credit Card'),

(10, '2024-02-06', 'Bank Transfer');

**1. Basic SQL Queries (Any three)**

1. Retrieve all customer details.
2. Find customers who have placed more than two orders.
3. Get the total number of products in each category.
4. Identify the customers who have never placed an order.
5. Find the most expensive product in each category.

**2. SQL Joins (Any three)**

1. List all orders along with customer names.
2. Get the total amount spent by each customer.
3. Find the products purchased in each order.
4. Retrieve the order details along with product names and prices.
5. Show payment details for each order.

**3. Triggers (Any one)**

1. Prevent insertion of an order with a zero amount.
2. Log updates in an audit table when an order is modified.

**4. Subqueries (Any two)**

1. Retrieve customers who spent more than the average order value.
2. Find products that have never been ordered.
3. Identify the customer who placed the earliest order.

**5. Views**

1. Create a view that displays customer orders with their total amount.
2. Build a view that shows high-value transactions (above $1000).

**6. Stored Procedures**

1. Write a stored procedure that retrieves all orders for a given customer.
2. Implement a procedure to insert a new order with order details.

**7. Window Functions (Any Two)**

1. Rank customers based on total spending.
2. Find total revenue generated each month.
3. Identify the top-selling product per category.